CSE 250 Recitation

October 27 - 28: Graph Traversals

Traversal - DFS and BFS

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Discussion:

What ADT do we use for the TODO to do a **Depth First Traversal?**

What about for a **Breadth First Traversal?**

Traversal - DFS and BFS

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Discussion:

What ADT do we use for the TODO to do a **Depth First Traversal?** **Stack**

What about for a **Breadth First Traversal?** **Queue**

Traversal - BFS with **EdgeList**

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Exercise:

Do a breadth first traversal on the graph to the right, started at vertex **S**

Use your paper to track visited vertices, the state of TODO, and the edgeTo map.

EdgeList

Vertices:

A,B,C,D,E,F,G,H,S

Edges:

(A,S),(S,A),(C,E),(H,G),(A,C),
(H,E),(B,C),(C,D),(S,B),(F,B),
(D,H),(E,F),(E,H),(A,D),(C,A),
(F,G)

DO NOT draw the graph or convert to adjacency list. Use the Edge List!

Traversal - BFS with **EdgeList**

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
   a. Remove a node from the [TODO]
   b. For each of that nodes unvisited neighbors:
      i. Mark the neighbor as visited
      ii. Add it to our [TODO]

Discussion:

After the traversal we have the edgeTo map to the right

What path was found from S to H?

What about S to F?

edgeTo map:

```
A: (S,A)
B: (S,B)
C: (A,C)
D: (A,D)
E: (C,E)
F: (E,F)
G: (H,G)
H: (D,H)
```

Traversal - BFS with **EdgeList**

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
   a. Remove a node from the [TODO]
   b. For each of that nodes unvisited neighbors:
      i. Mark the neighbor as visited
      ii. Add it to our [TODO]

Discussion:

After the traversal we have the edgeTo map to the right

What path was found from S to H? S->A->D->H

What about S to F? S->A->C->E->F

edgeTo map:

```
A: (S,A)
B: (S,B)
C: (A,C)
D: (A,D)
E: (C,E)
F: (E,F)
G: (H,G)
H: (D,H)
```

Traversal - BFS with **EdgeList**

Doing the traversal on an EdgeList was kind of a pain…

Exercise:
Convert the EdgeList on the right to an AdjacencyList representation

You can write it out as a map. You only need to include outgoing edges.

After you create the AdjacencyList, write out pseudocode to explain how you converted it

EdgeList

Vertices:

A,B,C,D,E,F,G,H,S

Edges:

(A,S),(S,A),(C,E),(H,G),(A,C),
(H,E),(B,C),(C,D),(S,B),(F,B),
(D,H),(E,F),(E,H),(A,D),(C,A),
(F,G)

DO NOT draw the graph or convert to adjacency list. Use the Edge List!

Traversal - BFS with Adjacency List

EdgeList

Vertices:

A,B,C,D,E,F,G,H,S

Edges:

(A,S),(S,A),(C,E),(H,G),(A,C),
(H,E),(B,C),(C,D),(S,B),(F,B),
(D,H),(E,F),(E,H),(A,D),(C,A),
(F,G)

AdjacencyList

S : (S,A),(S,B)
A : (A,S),(A,C),(A,D)
B : (B,C)
C : (C,A),(C,E),(C,D)
D : (D,H)
E : (E,F),(E,H)
F : (F,B),(F,G)
H : (H,E),(H,G)

Discussion: Would doing the BFS traversal now be easier? Why?

Two Possibilities

Option 1

```
adjList = {}

for e in edges:

  adjList[e.from] += e
```

Option 2

```
adjList = {}

for v in vertices:

  for e in edges:

    if e.from == v:

      adjList[e.from] += e
```

Discussion: Who used option 1? 2? What is the complexity of each?

Two Possibilities

Option 1: $\Theta(m)$

```
adjList = {}
for e in edges:
  adjList[e.from] += e
```

Option 2: $\Theta(n \cdot m)$

```
adjList = {}
for v in vertices:
  for e in edges:
    if e.from == v:
      adjList[e.from] += e
```

Discussion: Who used option 1? 2? What is the complexity of each?

Traversal - Dijkstra's

Exercise:
Draw the graph.

Add edge weights to each edge.

Use weights that would cause the shortest path from **S** to **H** to be different than the path with the fewest edges.

AdjacencyList

```
S : (S,A),(S,B)
A : (A,S),(A,C),(A,D)
B : (B,C)
C : (C,A),(C,E),(C,D)
D : (D,H)
E : (E,F),(E,H)
F : (F,B),(F,G)
H : (H,E),(H,G)
```

Traversal - BFS/DFS vs Dijkstra's

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Discussion:

What should our TODO list be to perform Dijkstra's algorithm to find shortest paths?

Does the above traversal work for Dijkstra's?

Traversal - BFS/DFS vs Dijkstra's

Traversal Algorithm:

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Discussion:

What should our TODO list be to perform Dijkstra's algorithm to find shortest paths? **PriorityQueue**

Does the above traversal work for Dijkstra's? **No!**

It marks vertices as visited before we know if we found them by the shortest path!

Traversal - BFS/DFS vs Dijkstra's

Traversal Algorithm (DFS/BFS):

1. Insert the starting node into the [TODO] and mark as visited
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Traversal Algorithm (Dijkstra's):

1. Insert the starting node into the [TODO]
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. Mark the removed node as visited
    c. For each of that nodes unvisited
        i. Add it to our [TODO] with its distance from starting vertex

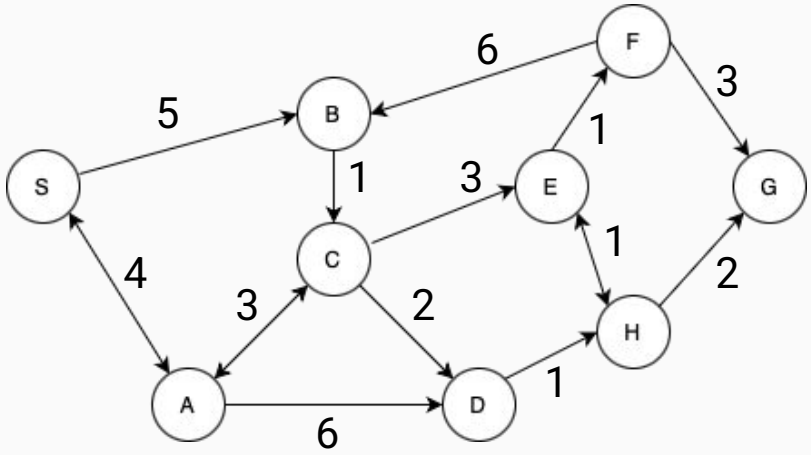
Exercise:

Trade papers with a neighbor and perform a full Dijkstra's traversal on their drawn graph. Use your paper to track visited vertices, the state of TODO, and the edgeTo map.

Check a few of the paths discovered to verify they are the shortest.

If you finish early, try the BFS algorithm with a PriorityQueue to verify for yourself that it does not work.

Traversal - Dijkstra's using AdjList



Traversal Algorithm (Dijkstra's):

1. Insert the starting node into the [TODO]
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. Mark the removed node as visited
    c. For each of that nodes unvisited
        i. Add it to our [TODO] with its distance from starting vertex